CPSC 350 – Data Structures
Spring 2019
Assignment #3: Syntax Checker
Due: 3-26-19, 11:59pm

Overview:

A major annoyance when programming is trying to figure out if you have put the correct number of delimiters (parenthesis, brackets) in a piece of code. This usually leads to trying to manually count the number of opening and closing delimiters to make sure everything matches up. Your assignment for this week is to write a C++ program that uses a stack to analyze source code written in any language (C, C++, Lisp…you name it) and report the location of any mismatched delimiters. For this assignment, delimiters are defined to be: ( ) { } [ ] Note that the assignment is not asking you to find any other type of syntax errors. (You can tackle that in a compilers class.)

The Specifics:

1. Implement your own stack class based on arrays. To avoid naming conflicts, call it GenStack. It should implement all the operations one would expect to see for this data structure. Your stack class must be generic (meaning it can hold any type of data – think templates), and it must automatically allocate more room for itself when it fills up. If the pop operation is called when the stack is empty, you should throw an exception. Write your code in such a way that it can be reused in future projects. This portion of the assignment will count the most.

2. 2. Write a program that takes the location of a source code file to be analyzed as a command line argument. (eg. ./assign3 foo.cpp).

3. 3. If the delimiters in the file are ok, report to the user that the delimiter syntax is correct, and ask if they want to analyze another file.

4. 4. If there is a problem, you must tell the user the location of the problem, as well as what the problem is. (eg. "Line 20: expected) and found]" or "Reached end of file: missing }") The program should then exit so as to allow the user to fix the error. (In other words, your program can just find 1 error at a time. This should make your life easier.)

The Rules:

1. You must work individually
2. All code submitted **MUST** be your own
3. Your stack implementation must be coded from scratch. Feel free to use whatever books you like for reference, but please cite them in a README.
4. You may develop on any platform you wish, but make sure your program compiles and runs the way you want it to using g++ within the Linux terminal

Deliverables:

Submit ONLY your commented source code and support files to github.

Due Date:
This assignment is due at 11:59pm on 3-26-19.

Grading:
Grades will be based primarily on correctness, adherence to the above guidelines, and elegance of implementation (an OO, modular design as opposed to procedural spaghetti code). Documentation and coding style will also contribute to your final grade.

Code Organization:
Your stack implementation and delimiter matching program should be implemented in separate files for the sake of reusability. You should have a GenStack.h file and a main.cpp file